Transformación de modelos aplicada a la definición genérica de Casos de Uso utilizando QVT (Query/View/Transformation) y RTG (Reglas de Transformación de Grafos)

Marcela Daniele, Ariel Arsaute, Mariana Frutos, Ariel Gonzalez, Paola Martellotto, Marcelo Uva, Fabio Zorzan, Daniel Riesco[1]

Universidad Nacional de Río Cuarto, Facultad de Ciencias Exactas, Físico-Químicas y Naturales, Departamento de Computación
Ruta 36 Km. 601 - Río Cuarto - Córdoba Tel. 0358-4676235
{marcela, aarsaute, mfrutos, agonzalez, paola, uva, fzorzan}@dc.exa.unrc.edu.ar

[1] Universidad Nacional de San Luis
Ejercito de Los Andes 950 CP D5700HHW
San Luis - San Luis - Argentina 54+2652+424027 int 251 driesco@unsl.edu.ar

Resumen

Este trabajo realiza un aporte tendiente a la mejora del proceso de desarrollo de software, siguiendo la filosofía de Arquitectura Dirigida por Modelos (Model Driven Architecture (MDA)). Las Plantillas Genéricas fueron definidas dentro de esta línea de investigación con el objetivo de proporcionar una aplicación práctica y estandarizada a procesos de desarrollo basados en casos de uso permitiendo plantear soluciones genéricas, estandarizadas y validadas a problemas similares, en las etapas de captura de requisitos, análisis y diseño. En este trabajo se presenta un meta-modelo para la definición de las Plantillas Genéricas utilizadas durante la captura de requisitos y se propone una transformación de los modelos instanciados a este nuevo meta-modelo a través de la aplicación de QVT (Query/View/Transformation) y Reglas de Transformación de Grafos, aprovechando los beneficios de estas técnicas y realizando un aporte hacia la automatización de aquellos procesos de desarrollo dirigidos por casos de uso.

Palabras clave: MDA, QVT, Reglas de Transformación de Grafos, Proceso Unificado.

Contexto

La línea de investigación presentada en este trabajo se desarrolla en el marco del proyecto "Ingeniería de Software: Automatización de Procesos de Desarrollo de Software", perteneciente a los Proyectos y Programas de Investigación (PPI) de la secretaría de Ciencia y Técnica de la Universidad Nacional de Río Cuarto.

Introducción

La Ingeniería de Software implementa un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software. Las metodologías de desarrollo junto con las herramientas de software se han adoptado con éxito en un amplio espectro de aplicaciones industriales. Dentro de las metodologías orientadas a objetos, el Proceso Unificado [1] define actividades y responsabilidades estableciendo quién está haciendo qué, cuándo, y cómo para construir o mejorar un producto de software. Esta metodología, divide el desarrollo del producto de software en fases utilizando UML (Unified Modeling Language) como lenguaje de modelado [2].

1.1 PLANTILLAS GENÉRICAS

En el año 2004, se desarrolló el proyecto "*Definición y uso de Plantillas Genéricas para la descripción de Casos de Uso*" [3], con el objetivo de proporcionar una aplicación práctica y estandarizada de la metodología de desarrollo del Proceso Unificado, permitiendo plantear soluciones genéricas, estandarizadas y validadas a problemas similares, en las etapas de captura de requisitos, análisis y diseño. Se propuso en ese momento una serie de plantillas genéricas para la descripción de casos de uso. La figura 1 muestra la plantilla para la inserción de elemento. En el año 2005, se desarrolló el proyecto "*Evolución de Plantillas Genéricas para la descripción de Casos de Uso a Plantillas para Análisis y Diseño"* [4], en donde se definieron las plantillas para las etapas de Análisis y Diseño, cubriendo las tres etapas fundamentales para una definición completa de la solución del problema a modelar.

Para la etapa de Diseño, se definió una plantilla genérica haciendo uso de patrones de diseño. Fueron utilizaron los patrones Mediator y Registry [5]. La Figura 2 muestra la platilla de diseño correspondiente al caso de uso de inserción, modificación, eliminación y búsqueda de un elemento. Basados en que la definición de soluciones genéricas, la plantillas producen una sustancial mejora en el accionar profesional en cuanto a que permiten plantear soluciones a problemas que requieren igual tratamiento que otros ya presentados, analizados y resueltos. La utilización de estas plantillas ha permitido reducir ampliamente la cantidad de documentación producida, facilitando la lectura y el entendimiento de cada uno de los casos de uso y definiendo soluciones estándar que favorezcan la comunicación entre los miembros del equipo de desarrollo. Por otro lado reduce la dificultad que se genera para comprender diversos modelos de solución planteados para resolver el mismo tipo de problemas, posibilitando destinar mayor cantidad de recursos a aquellos problemas que proporcionen un mayor valor agregado al sistema.

1.2 ARQUITECTURA DIRIGIDA POR MODELOS

La Arquitectura Dirigida por Modelos ("Model Driven Architecture (MDA)) [8,9] se ha concebido para dar soporte a la ingeniería dirigida por modelos de los sistemas software, cuyo objetivo central es resolver el problema producido por el cambio de tecnologías junto con la

Plantilla 1: Inserción de **<<elemento>>**

Parámetros:

Elemento: ítem a ser insertado. Un ítem está compuesto por atributos clave y atributos.
Atributos clave: las propiedades que identifican al elemento unívocamente.
Atributos: propiedades que componen el elemento.
Reglas de negocio ($r_1,..,r_n$): indicar las reglas que deben ser verificadas en el caso de uso.

Nombre: Inserción de <<elemento>>.
Pre-condición: existe un <<elemento>> a ser ingresado.
Post-condición: <<elemento>> queda registrado en el sistema, o <<elemento>> ya estaba registrado.
Descripción: realiza la inserción de un <<elemento>>, controlando la existencia del elemento en el sistema y el cumplimiento de las reglas del negocio ($r_1,..,r_n$) asociadas al <<elemento>>.
Actor: nombre de cada uno de los actores que interactúan con el caso de uso.

FLUJO DE EVENTOS PRINCIPAL	
ACTOR	SISTEMA
1. Ingresa <<atributos clave>> del <<elemento>>.	2. Verifica existencia por <<atributos clave>>.
3. Ingresa el resto de los <<atributos>> del <<elemento>>.	4. Verifica corrección de <<atributos>> ingresados. 5. Verifica reglas de negocio <<$r_1,..,r_n$>> asociadas al caso de uso. 6. Realiza el alta del <<elemento>>
FLUJO DE EVENTOS ALTERNATIVO	
2.1. El sistema informa de la existencia del <<elemento>> identificado con <<atributos clave>>.	
4.1. El sistema informa que al menos uno de los <<atributos>> ingresado no es correcto[1].	
5.1. El sistema informa las reglas r_i que no se verifican (con $1 \le i \le n$).	

Figura 1: Plantilla Genérica para la Inserción de un elemento

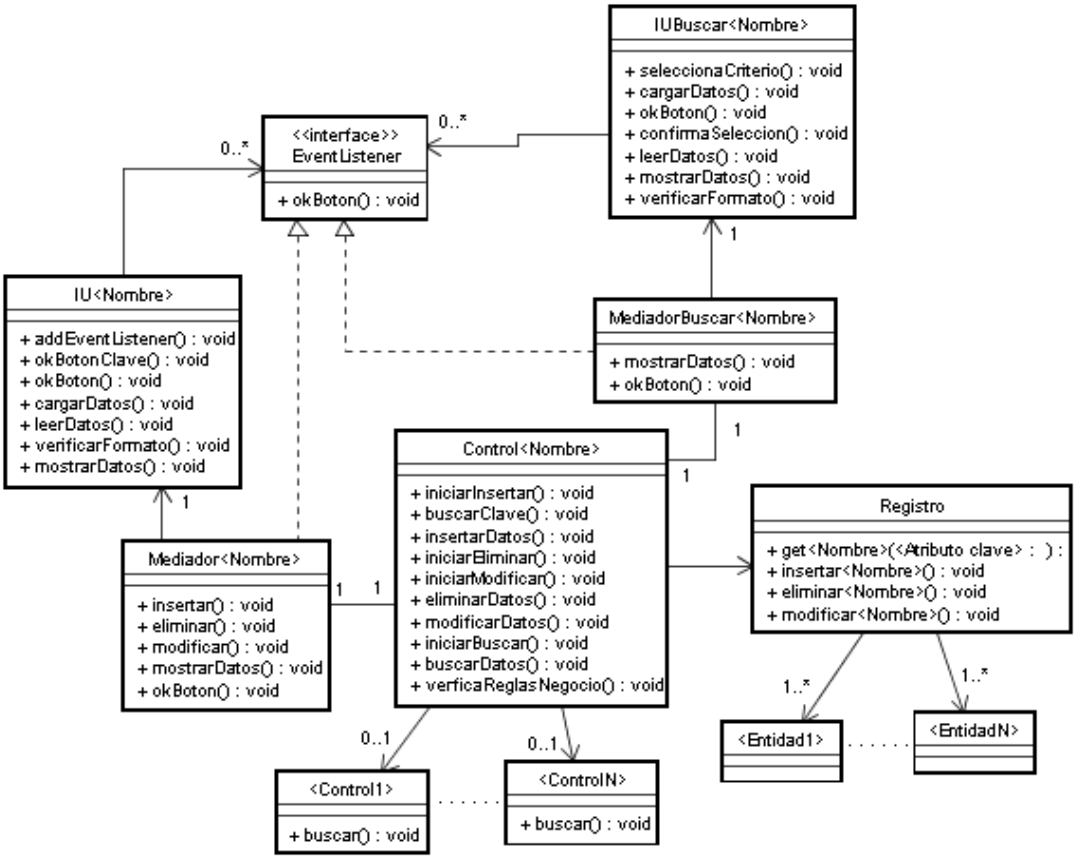

Figura 2: Plantilla Genérica para el Diseño

integración del sistema de software y que todo esto no impliquen un alto costo.

La idea subyacente en MDA es usar modelos, de modo que las propiedades y características de los sistemas queden contenidas en un modelo abstracto independiente de los cambios producidos en la tecnología. MDA proporciona una serie de guías o patrones expresadas como modelos. MDA propone cuatro niveles de abstracción que componen la jerarquía o arquitectura de modelos. Estos son: Modelo independiente de cómputo CIM (Computation Independent Model), Modelo independiente de la plataforma PIM (Platform Independent Model), Modelos específicos de la plataforma PSM (Platform Specific Model), y la aplicación final.

Los modelos CIM describen el entorno en el que se utilizará el sistema, sin referencia directa a su implementación.

Los PIM modelan funcionalidad y estructura de un sistema de información sin considerar detalles tecnológicos de la plataforma en la cual se implementará el sistema.

Los PSM describen los modelos específicos de plataforma, concretamente de la plataforma tecnológica donde se ejecutará el sistema.

MDA plantea el siguiente proceso de desarrollo: de los requisitos se obtiene un modelo independiente de la plataforma (PIM), luego este modelo es transformado con la ayuda de herramientas en uno o más modelos específicos de la plataforma (PSM), y finalmente cada PSM es transformado en código. Por lo tanto, MDA incorpora la idea de transformación de modelos (PIM a PSM, PSM a código) y se necesitan herramientas para automatizar estas tareas. Estas herramientas constituyen uno de los elementos básicos de MDA.

1.3 QVT (Query / View / Transformation)

El planteamiento QVT se basa principalmente en la definición de un lenguaje para las consultas (Queries) sobre los modelos Meta Object Facility (MOF) [6], la búsqueda de un estándar para generar vistas (Views) que revelen aspectos específicos de los sistemas modelados, y

finalmente, la definición de un lenguaje para la descripción de transformaciones de modelos MOF.

En este trabajo se plantea la utilización de QVT para definir transformaciones entre modelos. Estas transformaciones describen relaciones entre un metamodelo fuente F y un metamodelo objetivo O, ambos metamodelos deben estar especificados en MOF. Luego esta transformación se utiliza para obtener un modelo objetivo, el cual es una instancia del metamodelo O, a partir de un modelo fuente que es una instancia del metamodelo F. Una característica muy importante de estas transformaciones es que pueden ser bidireccionales y multidimencionales.

1.4 REGLAS DE TRANSFORMACIÓN DE GRAFOS

La transformación de grafos es una técnica gráfica, formal, declarativa y de alto nivel muy usada para transformación y simulación de modelos, chequeo de consistencia entre modelos o vistas y optimización en diversos contextos. Los artefactos producidos para conceptualizar un sistema son llamados modelos, y los diagramas son usados para visualizar su estructura compleja de una forma intuitiva y natural. Esta técnica formal para la manipulación de grafos está basada en la definición de reglas. Los modelos gráficos se interpretan como grafos y se utilizan dichas reglas para producir transformación entre permitiendo generar nuevos modelos.

Las técnicas de transformación de grafos se pueden aplicar para el modelado de requerimientos funcionales de un sistema de software (Casos de Uso). Permitiendo especificar los requerimientos de una manera visual, y a la vez formal, con los beneficios adicionales de una especificación ejecutable. Las reglas de transformación de grafos, proveen una especificación de alto nivel, en términos de pre y pos-condición, permiten además la especificación de la sintaxis abstracta y concreta junto con la semántica de los lenguajes de modelado visual, tanto para aspectos estáticos como los dinámicos. Es posible la utilización de sistemas de transformación de grafos para especificar la correspondencia entre la sintaxis abstracta y concreta y entre la semántica abstracta y concreta.

Líneas de investigación y desarrollo

Como se mencionó anteriormente, MDA propone basar el desarrollo de software en modelos, separando el diseño de la arquitectura y de las tecnologías de construcción, posibilitando que el diseño y la arquitectura puedan ser alterados independientemente. Una vez definidos los modelos, se definen una serie de trasformaciones que generan nuevos modelos [7], permitiendo ir de modelos más abstractos a otros más concretos. MDA define un framework para procesar y relacionar modelos.

La transformaciones entre modelos se realizan haciendo uso de herramientas automáticas, como herramientas de transformación de modelos, las cuales permiten el refinamiento de los mismos.

Nuestra propuesta consiste en una primera etapa, trabajar en la formalización de las Plantillas Genéricas para la descripción de casos de uso a través de un meta-modelo UML. En una segunda etapa, se tiene planificado la trasformación de modelos pre-establecidos en las plantillas genéricas utilizando QVT y reglas de transformación de grafos para la generación de los modelos correspondientes a las etapas de análisis y diseño. De esta manera se pretende lograr un acercamiento a la automatización de la construcción del software aplicada a aquellos procesos de desarrollo dirigidos por casos de uso.

Resultados y Objetivos

Se está trabajando en la definición de una primera versión del meta-modelo para la definición de las Plantillas Genéricas para la descripción de casos de uso. Al observar las características de la plantilla presentada en la figura 1 con el meta-modelo propuesto, se podrá establecer una clara relación instancia-modelo. El resultado esperado está estrechamente vinculado al logro de un acercamiento hacia la automatización de las actividades que componen el proceso de desarrollo de software, transformado modelos ya validados como lo son las Plantillas Genéricas de descripción análisis y diseño de casos de uso,

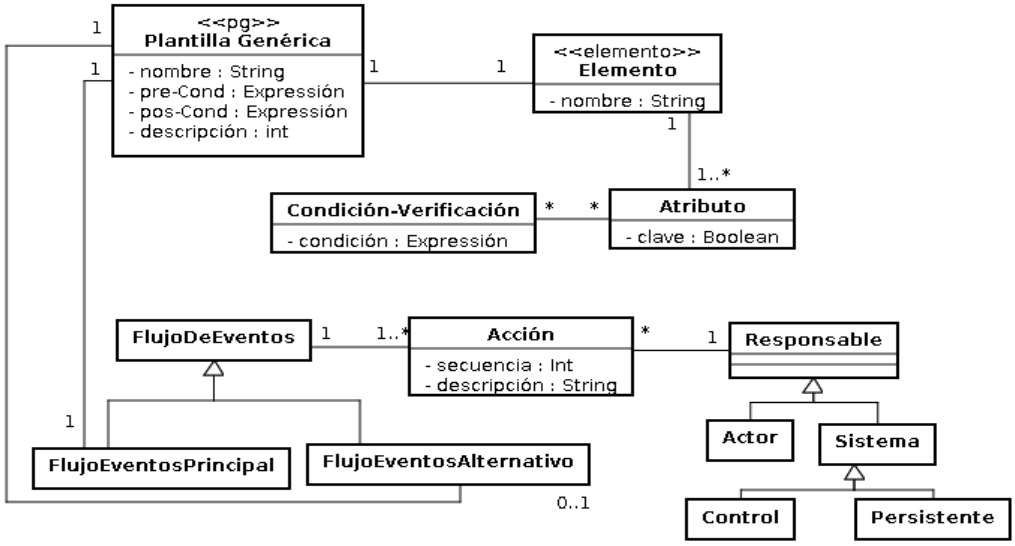

Figura 3: Meta-modelo de definición de Plantillas Genéricas para la descripción de casos de uso

con el fin de obtener un producto de software de calidad, cumpliendo con los requerimientos iniciales.

Formación de Recursos Humanos

Durante el desarrollo de esta línea de investigación han logrado obtener el título Magister en Ingeniería de Software tres integrantes del grupo de trabajo. Otro integrante está actualmente trabajando es su tesis de Magister. Dos grupo de alumnos están realizando su trabajo final de fin de carrera de Analista en Computación, otro grupo ha logrado obtener el título de Licenciado en Ciencias de la Computación. También, se han formado ayudantes de segunda en las asignaturas de Análisis y Diseño de Sistemas, Ingeniería de Software, Base de Datos y Proyecto.Los temas abordados en esta línea de investigación brindan un fuerte aporte al proceso de perfeccionamiento continuo de los autores de carreras de computación en Universidades Nacionales como del exterior.

Referencias

[1] Jacobson, I. El Proceso Unificado de Desarrollo de software. Addison-Wesley, EE.UU., 2000.

[2] UML, Unified Modeling Language (UML) Resource Page. http://www.omg.org/#UML2.0

[3] Daniele, M., Florio, N., Romero, D.. Definición y uso de Plantillas Genéricas para la descripción de Casos de Uso. PIIMEG 2004. UNRC.

[4] Daniele, M. y et al.ot.. Evolución de Plantillas Genéricas para la descripción de Casos de Uso a Plantillas para el Análisis y Diseño. PIIMEG 2005. UNRC.

[5] Gamma Erich y et. al. 1995. "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley

[6] Object Management Group, "Meta Object Facility (MOF)2.0Query/View/Transformation Specification" http://www.omg.org/docs/ptc/05-11-01.pdf.

[7] N. Debnath, F. A. Zorzan, G. Montejano and D. Riesco, "Transformation of BPMN Subprocesses Based in SPEM Using QVT", 2007 IEEE INTERNATIONAL CONFERENCE on ELECTRO/INFORMATION TECHNOLOGY, May 17-20, 2007, Marriott O Hare, Chicago, IL, USA. http://www.eit-conference.org/eit2007/

[8]. Miller, J., Mukerji, J., MDA Guide Version 1.0.1 Document number omg/2003-06-01, Disponible en: http://www.omg.com/mda, 2003.

[9] Object Management Group "Meta Object Facility (MOF) Core Specification" OMG http://www.omg.org/docs/formal/06-01-01.pdf.